**《花开福德系统》**

**项目进程报告**

**课程名称： 软件项目综合实践**

**指导教师： 梁琦**

北京信息科技大学

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 学号 | 团队名 | 完成日期 | 文件状态 | 文件标识 | 当前版本 |
| 王欣鹏 | 2021011352 | 北京信息科技大学食全食美责任有限公司 | 2024/10/18 | 完成 | 0001 | 1.0.0 |

## 1.引言：

### 编写目的：

向项目发起人、管理层和其他利益相关者传达项目的当前状态，包括已完成的工作、正在进行的任务以及未来的里程碑。通过比较实际进度与计划进度，实时评估项目是否按预定计划进行。报告中包含项目中遇到的问题、挑战和风险，以及针对这些问题的解决方案或缓解措施。有助于团队成员和管理层共同应对潜在问题。通过分享项目信息，促进团队成员之间的沟通和协作，确保每个人都对项目有共同的理解和期望。通过定期发布进展报告，增强项目的透明度，使所有利益相关者都能了解项目的最新动态。

### 背景：

待开发的软件系统的名称为《北京信息科技大学食全食美责任有限公司花开福德系统》（以下简称为：福德系统）。本项目的任务提出者为北京信息科技大学食全食美责任有限公司，开发地点：北京信息科技大学（沙河校区）XXB-402，联系电话：18511074814，用户为16-80岁的有动手能力，以及信息检索能力的人。该系统是一个独立的计算机系统，与现运行的其他系统无接口。

## 2项目进程情况：

### **2.1任务与进度：**

**用户注册与登录（进度80%）：**

提供用户注册功能，允许用户创建账户并设置个人信息。支持用户登录，通过验证用户名和密码来确保用户身份。

**设置:**

允许用户调节界面颜色，切换账号，退出登录等。

**美食列表与详情：**

展示系统推荐的美食列表，包括美食的名称、图片、简介等。提供美食详情页面，展示美食的详细介绍、制作步骤、食材清单等。

**评论区：**

允许用户对美食进行打分和评论，并展示其他用户的评价信息。

**收藏：**

如果您对这道菜感兴趣，您可以点击收藏，将可以在收藏界面看到您收藏的菜品.在收藏中会显示：食品名称，食品分类，收藏的时间等。

### 2.2费用：

**计划：**

**支出一览表（单位元）**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 功能点成本 | 人工成本 | 硬件和基础设施费用 | 软件许可工具费用 | 测试和维护费用 | 项目管理费用 | 市场营销和推广费用 | 总计 |
| 42872.45 | 300 /天 | 50000 | 20000 | 22767.25 | 11383.62 | 50000 | 401823.32 |

最后，我们加上风险准备金（假设为初步总预算的20%）：  
401823.32 \* 20% = 50364.66元（四舍五入到小数点后两位）。

因此，美食推荐系统项目的总预算估算为：  
401823.32 + 50364.66 = 452187.98元（四舍五入到小数点后两位）。

预算为500000元整，故

**最高收益**500000 - 401823.32 = 98176.68元

**最低收益**500000 - 452187.98 = 47812.02元

**平均收益**：72994.35

**实际情况：**

**支出一览表（单位元）**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 功能点成本 | 人工成本 | 硬件和基础设施费用 | 软件许可工具费用 | 测试和维护费用 | 项目管理费用 | 市场营销和推广费用 | 总计 |
| 22487.34 | 210 /天 | 20000 | 1000 | 8795.73 | 11383.62 | 50000 | 249283.07 |

最后，我们加上风险准备金（假设为初步总预算的20%）：  
249283.07 \* 20% = 49856.61元（四舍五入到小数点后两位）。

因此，美食推荐系统项目的总预算估算为：  
249283.07 + 49856.61 = 299139.68元（四舍五入到小数点后两位）。

预算为350000元整，故

**最高收益**350000 – 249283.07 = 100716.93元

**最低收益**350000 – 299139.68 = 50860.32元

**平均收益**：75788.63（四舍五入到小数点后两位）

### 2.3人力资源·：

**计划：**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 人员 | 管理组 | 开发组 | 质保组 | 配置组 | 用户组 |
| 武国璋 | 项目经理 |  |  |  | （兼） |
| 马天一 |  | 组长（前端设计） |  |  | （兼） |
| 王欣鹏 |  | 前端开发负责人 |  |  | （兼） |
| 王润淇 |  | 前端开发 |  |  | （兼） |
| 李昂 |  | 前端开发 |  |  | （兼） |
| 任品 |  | 后端开发负责人 |  |  | （兼） |
| 胡嘉乐 |  | 后端开发 |  |  | （兼） |
| 张子豪 |  | 后端开发 |  |  | （兼） |
| 赵圣杰 |  |  | 组长 |  | （兼） |
| 胡西贝 |  |  | 测试 |  | （兼） |
| 李羿霖 |  |  |  | 组长 | （兼） |

**实际情况：未更改**

### 2.4软硬件资源

**计划：**

#### 2.4.1系统架构概述

本系统是一个典型的 Python Web 应用程序，由多个模块组成，分为应用主文件、配置文件、扩展模块、数据模型和控制逻辑。这些模块通过 Flask 框架进行集成与协作，以提供一个功能完备且可扩展的 Web 后端服务。该系统采用 Flask 框架构建，模块化设计使得每个部分的职责清晰可见，易于维护和扩展。以下是对系统中各个模块的功能简要介绍：

1.app.py：主应用入口，负责创建应用实例并整合各个模块。

2.config.py：负责系统的配置管理，包括数据库、密钥等。

3.exts.py：用于初始化各类扩展，包括数据库和缓存等。

4.models.py：定义数据库模型，表示数据结构。

5.views.py：实现应用的业务逻辑和接口。

#### 2.4.2模块介绍

##### 2.4.2.1app.py

app.py是系统的主入口，包含创建 Flask 应用实例的代码。它负责整合所有其他模块并启动应用服务。

主要功能包括：

- 创建 Flask 应用实例。

- 加载配置（引用 `config.py`）。

- 初始化扩展（引用 `exts.py`）。

- 注册蓝图（引用 `views.py` 中定义的路由）。

通过这种设计，`app.py` 实现了对整个系统的初始化和协调。

##### 2.4.2.2config.py

config.py文件包含系统的配置信息，主要用于存储数据库连接字符串、密钥以及其他配置项。

配置项的作用包括：

- 开发环境和生产环境的区分：通过不同的配置类，例如 `DevelopmentConfig` 和 `ProductionConfig`，可以轻松切换环境。

- 数据库配置：提供数据库连接 URI，供扩展模块使用。

这使得系统配置变得灵活且集中，易于修改和管理。

##### 2.4.2.3exts.py

exts.py主要用于初始化系统中的各类扩展，如数据库、缓存等。通过这种方式，应用程序与扩展的初始化解耦，便于管理和测试。

- 数据库扩展：通常会使用 `SQLAlchemy` 或其他类似工具来管理数据库连接。

- 其他扩展：如缓存机制、用户认证等，也在这里进行初始化。

`exts.py` 确保了各类扩展在应用实例创建时得到正确初始化。

##### 2.4.2.4models.py

models.py负责定义系统的数据模型。使用 `SQLAlchemy` 等工具来描述数据库中的实体结构。

主要功能：

- 定义数据表结构：如用户表、订单表等。

- 提供数据库操作方法：如 CRUD（创建、读取、更新、删除）操作。

通过这种设计，`models.py` 把数据逻辑与业务逻辑分离，使得代码更具可读性与可维护性。

##### 2.4.2.5views.py

views.py包含应用的控制逻辑和路由定义，处理用户请求并返回相应的响应。

主要功能包括：

- 定义路由：使用 Flask 的装饰器定义每个 API 的路由，例如 `@app.route('/user')`。

- 处理请求逻辑：调用 `models.py` 中的数据操作方法，处理用户请求。

- 返回响应：根据请求结果返回合适的 JSON 数据或页面。

这种视图层的设计使得业务逻辑清晰明了，易于扩展和修改。

#### 2.4.3模块交互关系

- app.py是整个系统的核心，负责初始化应用和加载所有模块。

- config.py为系统提供配置支持，exts.py和views.py都依赖于它。

- exts.py负责初始化所有扩展，供models.py和views.py使用。

- models.py提供的数据模型被views.py使用，用于处理数据库中的数据。

这种模块化设计使得每个模块职责明确，减少了模块之间的耦合度，提升了系统的可维护性。

#### 2.4.4技术细节

##### 2.4.4.1应用实例初始化

在 `app.py` 中，Flask 应用实例通过如下步骤进行初始化：

1. \*\*创建 Flask 应用实例\*\*：

app = Flask(\_\_name\_\_)

2. \*\*加载配置\*\*：从 `config.py` 中加载相应的配置类：

app.config.from\_object('config.DevelopmentConfig')

3. \*\*初始化扩展\*\*：调用 `exts.py` 中的扩展初始化函数。

from exts import db

db.init\_app(app)

4. \*\*注册蓝图\*\*：将 `views.py` 中定义的蓝图注册到应用实例：

from views import main\_blueprint

app.register\_blueprint(main\_blueprint)

##### 2.4.4.2数据库配置与扩展

在 `config.py` 中，数据库连接配置使用 SQLAlchemy URI 的形式提供，例如：

SQLALCHEMY\_DATABASE\_URI = 'mysql+pymysql://user:password@localhost/dbname'

在 `exts.py` 中，数据库扩展通过如下方式进行初始化：

from flask\_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

然后在 `app.py` 中调用 `db.init\_app(app)` 以将数据库与应用实例关联。

##### 2.4.4.3视图和路由的实现

在 `views.py` 中，控制逻辑通过路由进行定义，例如：

from flask import Blueprint, request, jsonify

from models import User

main\_blueprint = Blueprint('main', \_\_name\_\_)

@main\_blueprint.route('/users', methods=['GET'])

def get\_users():

users = User.query.all()

return jsonify([user.to\_dict() for user in users])

这个视图函数负责处理 `/users` 路径的 GET 请求，调用 `models.py` 中的 `User` 模型来获取数据并返回 JSON 响应。

#### 2.4.5部署与运行

1. \*\*安装依赖\*\*：通过 `pip` 安装所有依赖包：

pip install -r requirements.txt

2. \*\*设置环境变量\*\*：设置 Flask 的环境变量以指定应用入口：

export FLASK\_APP=app.py

export FLASK\_ENV=development

3. \*\*运行应用\*\*：使用 Flask 内置服务器运行应用：

flask run

**实际情况：未更改**

## 3 问题与对策：

存在的问题

准确性问题：

数据质量：菜品相关数据的准确性和完整性直接影响推荐系统的准确性。如果数据存在错误或缺失，将导致推荐结果不准确。

算法模型：算法模型的选择和参数调整也是影响推荐准确性的关键因素。不同的算法模型适用于不同的场景和业务需求，如果选择不当或参数调整不合适，将影响推荐效果。

个性化程度不足：

美食推荐系统应该根据用户的喜好和行为习惯进行个性化推荐。然而，在实际应用中，由于用户数据的稀缺性或算法模型的局限性，往往难以实现高度个性化的推荐。

多样性问题：

推荐结果的多样性对于提升用户满意度至关重要。然而，在实际应用中，由于算法模型的限制或数据分布的不均衡，推荐结果往往过于单一，缺乏多样性。

冷启动问题：

对于新用户或新菜品，由于缺乏足够的历史数据，推荐系统往往难以提供准确的推荐。这就是所谓的冷启动问题。

解决方案

提高数据质量和算法模型：

餐厅应确保菜品相关数据的准确性和完整性，包括菜品的名称、描述、图片、价格等。同时，应选择合适的算法模型，并根据具体业务情况进行参数调整，以提高推荐的准确性。

增强个性化推荐：

通过收集用户反馈、建立用户画像和利用社交网络数据等方式，可以更深入地了解用户的喜好和行为习惯，从而实现更个性化的推荐。例如，可以基于用户的购买历史、浏览记录、点赞和评论等数据，构建用户画像，并根据画像进行精准推荐。

提升推荐结果的多样性：

可以通过融合多种推荐算法、引入随机因素或利用多样性优化算法等方式，增加推荐结果的多样性。例如，可以将基于内容的推荐、协同过滤推荐和深度学习推荐等算法进行融合，以提供更丰富多样的推荐结果。

解决冷启动问题：

对于新用户或新菜品，可以通过基于内容的推荐、利用用户反馈数据或激励用户产生行为数据等方式进行推荐。例如，可以向新用户推荐与其喜欢的菜品相似的菜品，或者通过给予用户一定的激励来鼓励他们主动产生行为数据，以便更好地了解他们的喜好并进行推荐。

|  |  |
| --- | --- |
| **存在问题** | **解决方案** |
| **数据质量** | 菜品相关数据的准确性和完整性直接影响推荐系统的准确性。如果数据存在错误或缺失，将导致推荐结果不准确。应确保菜品相关数据的准确性和完整性，包括菜品的名称、描述、图片、价格等。 |
| **算法模型** | 算法模型的选择和参数调整也是影响推荐准确性的关键因素。不同的算法模型适用于不同的场景和业务需求，如果选择不当或参数调整不合适，将影响推荐效果。应选择合适的算法模型，并根据具体业务情况进行参数调整，以提高推荐的准确性。 |
| **个性化不足** | 美食推荐系统应该根据用户的喜好和行为习惯进行个性化推荐。在实际应用中，由于用户数据的稀缺性或算法模型的局限性，往往难以实现高度个性化的推荐。通过收集用户反馈、建立用户画像和利用社交网络数据等方式，可以更深入地了解用户的喜好和行为习惯，从而实现更个性化的推荐。 |
| **多样性问题** | 推荐结果的多样性对于提升用户满意度至关重要。然而，在实际应用中，由于算法模型的限制或数据分布的不均衡，推荐结果往往过于单一，缺乏多样性。可以将基于内容的推荐、协同过滤推荐和深度学习推荐等算法进行融合，以提供更丰富多样的推荐结果。 |
| **冷启动问题** | 对于新用户或新菜品，由于缺乏足够的历史数据，推荐系统往往难以提供准确的推荐。对于新用户或新菜品，可以通过基于内容的推荐、利用用户反馈数据或激励用户产生行为数据等方式进行推荐。例如，可以向新用户推荐与其喜欢的菜品相似的菜品，或者通过给予用户一定的激励来鼓励他们主动产生行为数据，以便更好地了解他们的喜好并进行推荐。 |